A context semantics for interaction nets

Matthieu Perrinel ENS Lyon
Université de Lyon
LIP (UMR 5668 CNRS ENS Lyon UCBL INRIA
matthieu.perrinel@ens-lyon.fr

***Abstract*—Context semantics is a tool inspired by Girard' s geometry of interaction. It has had many applications from study of optimal reduction to proofs of complexity bounds. Yet, context semantics have been defined only on λ-calculus and linear logic.**

In order to study other languages, in particular languages with more primitives (built-in arithmetics, pattern matching,...) we define a context semantics for a broader framework: interaction nets. These are a well-behaved class of graph rewriting systems. It could be used to prove strong complexity bounds for functional languages.

I. Introduction

Context semantics (CS) is a tool related to geometry of interaction (GoI) [4], [8]. CS is a mean to study the evaluation of a program (a λ-term or a proof-net of linear logic) by means of paths in the program. Those paths are defined by a token travelling across the program according to some rules. It has first been used to study optimal reductions in λ-calculus [8] and linear logic [9]. Then, it has been used to prove complexity bounds on subsystems of System T [11] and linear logic [1], [3], [15]. An advantage of context semantics compared to the syntactic study of reduction is its genericity: some common results can be proved for different variants of linear logic, which allows to factor out proofs of complexity results for these various systems.

Since CS had many interesting developments in λ-calculus and linear logic, we would like to have a similar tool for programming languages. For instance, we want pattern-matching, inductive data-types (as opposed to Church encoding) and built-in arithmetics operation. As the set of features needed is not precisely defined, a general framework of systems would be preferred to a single system. This way, we would need to define the CS and prove the general theorems only once, and they will stand for any system of the framework. The framework we chose is interaction nets [10].

Interaction nets are a model of asynchronous deterministic computation. They are based on rewriting rules on graphs and were inspired by the proof-nets of linear logic [7]. Interaction nets can, in particular, encode proof-nets [13] and λ-calculus [12]. Moreover, interaction nets are general enough to encode functional programming languages containing pattern-matching and built-in recursion [6]. A non-deterministic extension is powerful enough to encode the full π-calculus [14].

A net is a graph-like structure whose nodes are called *cells*. Each cell is labelled by a symbol. A library defines the set of symbols and the rewriting rules for the symbols. Thus, a library corresponds to a programming language. Interaction nets as a whole, correspond to a set of programming languages.

Contributions: In this paper, we define CS for any library and we show that the CS paths are stable along reduction. For any net N, we define a weight $W_N \in \mathbb{N} \cup \{\infty\}$ based on CS paths. We prove that if M reduces to N, then $W_N = W_M - 1$. Thus, if N normalizes, W_N is the length of the reduction path, else $W_N = \infty$. This could be used to prove complexity bounds on programming languages which are either defined or encodable in interaction nets.

Related works: As CS is a model of GoI, the closest work to this paper, is the definition of a GoI for an arbitrary library by De Falco [5]. De Falco defines a notion of paths in nets and a notion of reduction of those paths. Then, he defines a GoI of a library as a weighing of paths by elements of a semi-group such that the weights are stable along reduction. However, one crucial lemma only stands for crossing libraries. Moreover, he exhibits such a semi-group only for some particular libraries (based on linear logic). Thus, there is no complete GoI model of interaction nets yet.

II. Interaction nets

A. Statics

Interaction nets have been defined in many ways. Here, to define properly the CS paths, we had to use a formal definition.

We fix a *symbol set* $S = (\mathcal{S}, \alpha)$ with $\mathcal{S}$ a countable set whose elements will be called *symbols* and α a mapping from $\mathcal{S}$ to $\mathbb{N}$ associating an *arity* to each symbol.

A net is a set of cells joined by wires. Wires may have one (or both) ends unattached. We will often connect nets, those connections are made by those unattached ends. Formally, the ends of wires will be represented by a set P^N of *ports*. There are three types of ports: ports attached to a cell (the set P_c^N), *free ports* (the set P_f^N) and *merging ports* (the set P_m^N). This latest group is used for technical reasons.

Definition 1. *A* net N *is a tuple* $(P^N, C^N, l^N, \sigma_w^N, \sigma_m^N, \sigma_c^N)$ *with:*

- $P^N = P_c^N \uplus P_f^N \uplus P_m^N$ *is a finite set called set of* ports.
- C^N *is a finite set whose elements will be called* cells
- $l^N : C^N \mapsto \mathcal{S}$ *labels each cell with a symbol.*
- σ_w^N *is an involution on* P^N *with no fixpoint. We also write* $\overline{p}$ *for* $\sigma_w^N(p)$. σ_w^N *represents the wires: if there is a wire between the ports* p *and* p'*, then* $\overline{p} = p'$ *and* $\overline{p'} = p$.
- σ_m^N *is an involution on* P_m^N *with no fixpoint. This mapping associates two merging ports.*
- σ_c^N *is a bijection from* P_c^N *to* $\{(c,i) | c \in C^N, 0 \leq i \leq \alpha(l^N(c))\}$. σ_c^N *represents the cells. For instance,*

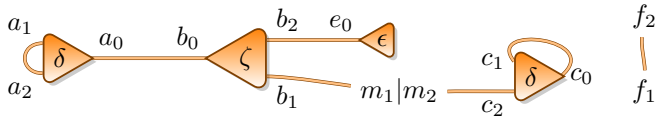


Fig. 1: Net N. Names of ports and labels of cells are represented while names of cells are not.

$\sigma_c^N(p) = (c, 2)$ *if* p *is the second auxiliary port of* c *and* $\sigma_c^N(p) = (c, 0)$ *if* p *is the principal port of* c.

For example, let $S_{comb} = \{\zeta, \delta, \epsilon\}$ be symbols with $\alpha(\zeta) = \alpha(\delta) = 2$ and $\alpha(\epsilon) = 0$. Then, Figure 1 represents the net N with: $P^N = \{a_0, a_1, a_2, b_0, b_1, b_2, c_0, c_1, c_2, e_0, f_1, f_2, m_1, m_2\}$, $C^N = \{A, B, C, E\}$, $l^N = \{A \mapsto \delta, B \mapsto \zeta, C \mapsto \delta, E \mapsto \epsilon\}$, $\sigma_w^N = \{a_1 \leftrightarrow a_2, a_0 \leftrightarrow b_0, b_2 \leftrightarrow e_0, b_1 \leftrightarrow m_1, m_2 \leftrightarrow c_2, c_1 \leftrightarrow c_0, f_1 \leftrightarrow f_2\}$, $\sigma_m^M = \{m_1 \leftrightarrow m_2\}$ and $\sigma_c^N = \{a_0 \mapsto (A, 0), a_1 \mapsto (A, 1), a_2 \mapsto (A, 2), b_0 \mapsto (B, 0), b_1 \mapsto (B, 1), b_2 \mapsto (B, 2), c_0 \mapsto (C, 0), c_1 \mapsto (C, 1), c_2 \mapsto (C, 2), e_0 \mapsto (E, 0)\}$.

The merging ports are introduced for technical reasons but are not essential. Let p, q be merging ports of a net N such that $p \neq \overline{q}$. Let N' be the net equal to N where $\overline{p} - p|q - \overline{q}$ is replaced by $\overline{p} - \overline{q}$, then we write $N \rightarrow_m N'$. We define the equivalence relation $\leftrightarrows_m$ as the reflexive symmetric transitive closure of $\rightarrow_m$. The nets will be considered up to $\leftrightarrows_m$ equivalence and α-equivalence (renaming of the ports and cells). Notice that $\rightarrow_m$ is confluent and strongly normalizing, we will usually represent a net by its $\rightarrow_m$ normal form (the only merging ports are the cycles of shape $p|q$).

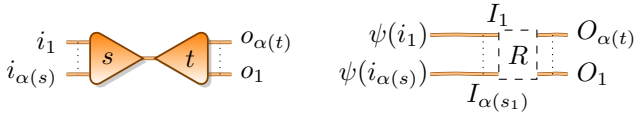
Let c be a cell of N. We write $p^i(c)$ the port p such that $\sigma_c^N(p) = (c, i)$. The *principal port* of c denotes $p^0(c)$. If $i \geq 1$, $p^i(c)$ is called the *i-th auxiliary port* of c.

B. Dynamics

The interaction between two nets is done by merging some of their free ports. This operation is called *gluing* and will be the main tool to define the dynamics of nets. Let M and N be nets and ϕ be a partial injection from P_f^M to P_f^N, then $M \bowtie_\phi N$ is the net whose ports and cells are those of M and N, the free ports in the domain and codomain of ϕ become merging nodes with $\sigma_m^{M\bowtie_\phi N}(p) = \phi(p)$ and $\sigma_m^{M\bowtie_\phi N}(\phi(p)) = p$. As an example, if we set $M =$  , $N =$  $f_1 - f_2$ and $\phi = \{m_1 \mapsto m_2\}$, then $M \bowtie_\phi N$ is the net of Figure 1.

The computation in interaction nets is done by reduction of *active pairs*. An active pair is a set of two cells linked by their principal ports. *Libraries* will define which pairs of symbols can interact. When an active pair is labelled by symbols which can interact together, we may reduce it: those cells are replaced by a net $N_{s,t}$ which only depends on the symbols of the active pair. The rest of the interaction net is left untouched.

Definition 2. *Let* $s, t \in \mathcal{S}$, $\mathfrak{R}_{s,t}$ *is the net of Figure 2a.*



(a) The net $\mathfrak{R}_{s,t}$ (b) The reduct $N_{s,t}$ of s/t

Fig. 2: Interaction rule with explicit bijection ($O_k = \psi(o_k)$).

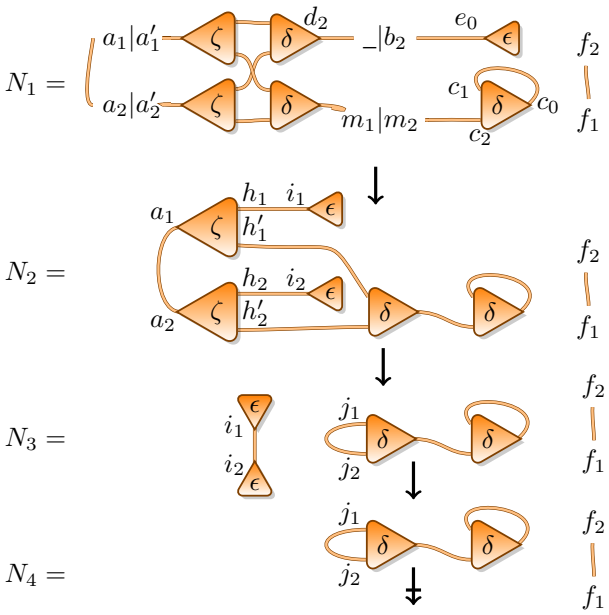


Fig. 4: Example of reduction with the library L_{comb}.

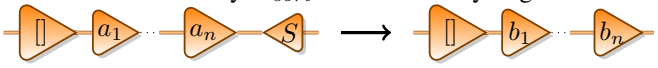
An interaction rule for (s, t) *is a tuple* (R, ψ) *where* R *is a net and* ψ *is a bijection from* $P_f^{\mathfrak{R}_{s,t}}$ *to* P_f^R. *For* $1 \leq j \leq \alpha(s)$, *we name* I_j *the port* $\overline{\psi(i_j)}$ *of* R. *For* $1 \leq j \leq \alpha(t)$, *we name* O_j *the port* $\psi(o_j)$ *of* R, *as in Figure 2b.*

In practice, we will describe interaction rules by displaying an active pair and the reduct linked by an arrrow as in Figure 3. The bijection is given implicitly by the position of the ports.

Definition 3 (library)**.** *A* library *for the symbol set* $(\mathcal{S}, \alpha)$ *is a partial mapping* L *on* $\mathcal{S} \times \mathcal{S}$. *To each* (s_1, s_2) *in the domain of* L, L *associates an interaction rule for* (s_1, s_2). *Let us suppose that* $L(s_1, s_2) = (R, \psi)$. *Then we require that* $L(s_2, s_1)$ *is defined and equal to the* symmetric *of* $L(s_1, s_2)$ *where inputs and outputs are switched, i.e.* $L(s_2, s_1) = (R, \psi \circ \{i_k \leftrightarrow o_k\})$. *The reduction* $\rightarrow$ *is defined by* $N \bowtie_\phi \mathfrak{R}_{s_1,s_2} \rightarrow (N \bowtie_{\psi \circ \phi} R)$.

Because of the symmetry condition, the rules shown in Figure 3 are enough to describe the whole library L_{comb} of symmetric combinators. The net of Figure 1 successively reduces to the nets of Figure 4 (note that we use the notation _ to denote an object whose name and value has no importance).

As another example, let us consider an ordered set $(A, \leq)$, and the symbols $\{S, []\} \cup A \cup \{I_a | a \in A\} \cup \{\overline{a} | a \in A\}$. The arities and the library L_{sort} are defined by Figure 5. Then,



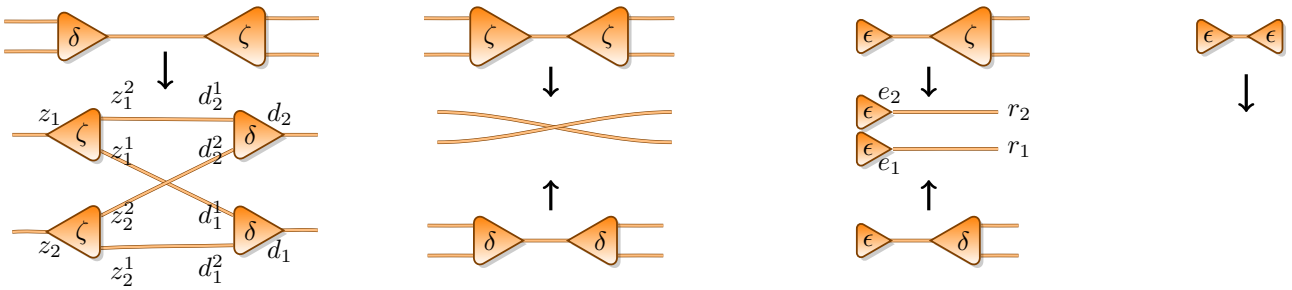


Fig. 3: The symmetric combinators, library L_{comb} for S_{comb}

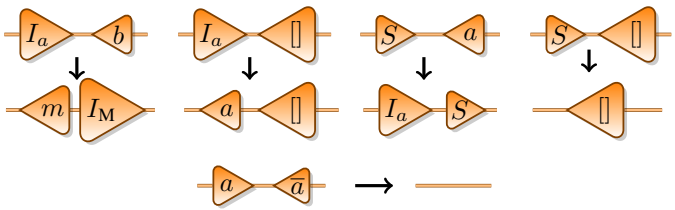


Fig. 5: L_{sort} library. The rules stand for any $a, b \in A$, $m = \min(a, b)$ and $M = \max(a, b)$

with $[b_1; \cdots ; b_n]$ the sorted list corresponding to $[a_1; \cdots ; a_n]$. More precisely, it is an implementation of insertion sort.

III. Context semantics

In this section, we fix a library L. For any (s_1, s_2) in the domain of L, we write $(N_{s_1,s_2}, \phi_{s_1,s_2}) = L(s_1, s_2)$. This section uses many lists. Lists are written in the form $[a_1; \cdots ; a_n]$, $l_1@l_2$ represents the concatenation of l_1 and l_2, . represents "push" ($[a_1; \cdots ; a_n].b = [a_1; \cdots ; a_n; b]$) and $|l|$ is the length of l.

Let us consider, in the net N_3 of Figure 4, the wire from i_1 to i_2. The wire appears in N_3 because the interaction of the two ζ cells in N_2 creates a wire between the second auxiliary ports of the cells. So, intuitively, the path $P_3 = i_1, i_2$ in N_3 comes from $P_2 = i_1, h_1, a_1, a_2, h_2, i_2$ in N_2. The path P_1 in N_1 corresponding to P_2 seems harder to define because the extremities i_1 and i_2 are not in N_1, they correspond to the ports e_1 and e_2 of $N_{\delta,\epsilon}$. We could represent the port i_1 in N_1 by the stack $[(e_0, N); (e_2, N_{\delta,\epsilon})]$, remembering what kind of port is created (e_2) and by which interaction it is created (e_0).

It is exactly the idea of our context semantics. The ports which will appear along the reduction are characterized by stacks called *potential ports* (e.g. $[(e_0, N); (e_2, N_{\delta,\epsilon})]$). The reduction is simulated by paths, which are defined by *contexts* travelling across the net according to a relation $\mapsto$. The contexts are pairs of a potential port P and a *trace* T. The trace T_i represents information about the beginning of the path.

The set Pot^N of *potential ports* of net N is the set of lists $[(p_0, N); (p_1, N_{s_1,t_1}); \cdots ; (p_k, N_{s_k,t_k})]$ such that for each i: p_i is a port of N_{s_i,t_i} and p_{i-1} is the principal port of a cell labelled by t_i. For $P \in Pot^N$, we set $\overline{P.(p, N')} = P.(\overline{p}, N')$.

A *positive trace element* is (s, i) with $s \in \mathcal{S}$ and $1 \leq i \leq \alpha(s)$. The meaning of (s, i) is "I have crossed a cell of symbol s, from its i-th auxiliary port to the principal port". A positive trace is a list of positive trace elements.

A *negative trace element* is $\overline{(s, i)}$ with $s \in \mathcal{S}$ and $1 \leq i \leq \alpha(s)$. The meaning of $\overline{(s, i)}$ is "I will arrive at the principal port of a cell of symbol s. When this happens I will choose to leave it by its i-th auxiliary port". A *trace element* is either a positive trace element or a negative trace element. A *trace* is a list of trace elements. The set of traces is written Tra.

We define the set of contexts of N_1 by $Cont^{N_1} = Pot^{N_1} \times Tra$. The intuitive meaning of a context $([(p_1, N_1); \cdots ; (p_k, N_k)], [(t_1, i_1); \cdots ; (t_l, i_l)])$ is: "N_1 reduces to a net of the shape of Figure 7 where for $1 \leq j \leq l$, q_j is the i_j-th auxiliary port of the cell labelled by t_j".

Thus, a potential port is a nesting of interaction nets: in the net N_1 there is a cell which will be part of an active pair and will produce the interaction net N_2 when reducing. Inside this N_2, there is a cell which will be part of an active pair and will produce N_3 when reducing, and so on.

Definition 4. *For any net N, we define a relation $\mapsto$ on $Cont^N$ by the rules of Figure 6. In those rules, we suppose $s, s' \in \mathcal{S}$, $c, c' \in C^N$, $l^N(c) = s$, $l^N(c') = s'$, $1 \leq k \leq \alpha(s)$, $1 \leq k' \leq \alpha(s')$ and $m, m' \in P_m^N$ with $\sigma_m^N(m) = m'$.*

This relation simulates reduction, in the sense that if $([(p, N)], []) \mapsto^* (P, T)$ and Figure 7 is the net intuitively representing (P, T), then $q_1 = p$. The $\mapsto$ relation is deterministic and incomplete (there are contexts C such that $C \not\mapsto$, i.e. $\forall D \in Cont^N, \neg(C \mapsto D)$). Let $C = (P, T) \in Cont^N$, the possible context D such that $C \mapsto D$ is defined depending on the right-most port p of P.

If p is an auxiliary port, we cross the cell and add the information on the trace (rule a).

If p is a principal port, the behaviour depends on whether the right-most trace element t is positive or negative (if the trace is empty, $C \not\mapsto_L$): if t is positive (rule c), then $t = (s, k)$ it corresponds to an active pair $\{c, c'\}$ of symbols $\{s, s'\}$. During reduction $N_{s,s'}$ will be glued to N, here we jump to $N_{s,s'}$ to simulate reduction. Else if t is negative (rule b), then $t = \overline{(s, k)}$. It means that at some moment in the path we crossed the cell c from its auxiliary port to its principal port. Then, we found a cell c' such that $\{c, c'\}$ will be an active pair. The $\mapsto$-path led us to a $\overline{I_k}$ free port. This port will correspond to the k-th auxiliary port of c during reduction, so we pushed $\overline{(s, k)}$ on the trace and took the path backward from c' to c so that we can reach the k-th auxiliary port of c.

If p is free, we are in the net $N_{s,s'}$ corresponding to the interaction of the future active pair $\{c, c'\}$. The behaviour

$$
\begin{array}{llllll}
a) & (P.(p_k(c), N) & ,T &) \mapsto & (P.(\overline{p_0(c)}, N) & ,T.(s,k)) \\
b) & (P.(p_0(c), N) & ,T.\overline{(s,k)} &) \mapsto & (P.(\overline{p_k(c)}, N) & ,T) \\
c) & (P.(p_0(c'), N) & ,T.(s,k) &) \mapsto & (P.(p_0(c'), N).(I_k, N_{s,s'}) & ,T) \\
d) & (P.(p_0(c'), N).(O_{k'}, N_{s,s'}) & ,T &) \mapsto & (P.(\overline{p_{k'}(c)}, N) & ,T) \\
e) & (P.(p_0(c'), N).(\overline{I_k}, N_{s,s'}) & ,T &) \mapsto & (P.(\overline{p_0(c)}, N) & ,T.\overline{(s,k)}) \\
f) & (P.(m, N) & ,T &) \mapsto & (P.(\overline{m'}, N) & ,T)
\end{array}
$$

Fig. 6: Rules of context-semantics

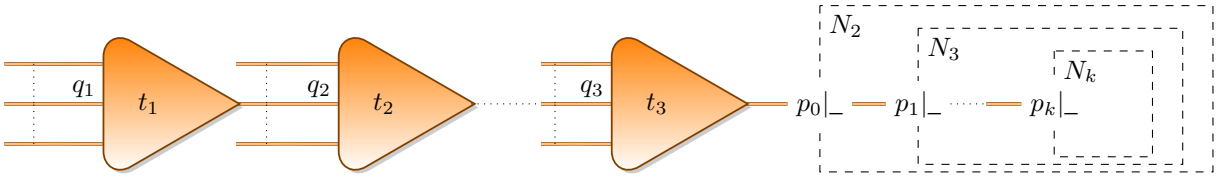


Fig. 7

depends on whether p is a $O_{k'}$ (rule d) or a $\overline{I_k}$ (rule e). In the first case, we must go to the k'-th auxiliary port of c' and it is local, in the second we have to go back to the cell c.

If p is a merging port, we cross the merging port (rule f).

As an example, the following $\mapsto$-path in the net N of Figure 1, goes from the principal port of E to the δ cell which will form an active pair with E. Notice that this δ cell does not exist yet (it will be created by the ζ/δ reduction): $([(b_2, N)], []) \mapsto ([(a_0, N)], [(\zeta, 2)]) \mapsto ([(a_0, N); (I_2, R_{\zeta,\delta})], [])$.

As a more involved example, we will study in the net N, the path between the two ϵ cells created during the δ/ϵ step of reduction (first step of Figure 4).

$$
\begin{array}{l}
([(e_0, N); (r_1, R_{\delta,\epsilon})], []) \mapsto ([(b_2, N)], [\overline{(\delta, 1)}]) \\
\mapsto ([(a_0, N)], [\overline{(\delta, 1)}; (\zeta, 2)]) \mapsto ([(a_0, N); (d_2, R_{\zeta,\delta})], [\overline{(\delta, 1)}]) \\
\mapsto ([(a_0, N); (z_1^2, R_{\zeta,\delta})], []) \mapsto ([(a_0, N); (O_1, R_{\zeta,\delta})], [(\zeta, 2)]) \\
\mapsto ([(\overline{a_1} = a_2, N)], [(\zeta, 2)]) \mapsto ([(b_0, N)], [(\zeta, 2); (\delta, 2)]) \\
\mapsto ([(b_0, N); (z_2, R_{\delta,\zeta})], [(\zeta, 2)]) \\
\mapsto ([(b_0, N); (z_2, R_{\delta,\zeta}); (I_2 = O_2, R_{\zeta,\zeta})], []) \\
\mapsto ([(b_0, N); (d_2^2, R_{\delta,\zeta})], []) \mapsto ([(b_0, N); (\overline{d_2}, R_{\delta,\zeta})], [(\delta, 2)]) \\
\mapsto ([(e_0, N)], [(\delta, 2)]) \mapsto ([(e_0, N); (e_2, N)], [])
\end{array}
$$

We wrote that $\mapsto$ simulates the reduction of the net. We will prove that the $\mapsto$-paths are stable by reduction. Formally, if $N \to N'$, we will define a projection Π from the potential ports of N to potential ports of N' so that $(P, T) \mapsto^* (Q, U) \Leftrightarrow (\Pi(P), T) \mapsto^* (\Pi(Q), U)$.

In this section, we suppose that $N \to N'$ by reducing the active pair $\{c_1, c_2\}$ labelled by s_1, s_2. We set $(R_1, \phi_1) = L(s_2, s_1)$ and $(R_2, \phi_2) = L(s_1, s_2)$. So $N = N_0 \bowtie_{\phi_2} \mathfrak{R}_{s_1,s_2}$ and $N' = N_0 \bowtie_{\psi \circ \phi_2} R_2$. We define a mapping Π from Pot^N to $Pot^{N'}$ which depends on the left-most port p:

- If $p \in P^{N_0}$, we set $\Pi([(p, N)]@P) = [(p, N')]@P$.
- If $p = p_0(c_i)$ for $i \in \{1, 2\}$. We set:

$$\Pi([(p_0(c_i), N); (r, R_i)]@P) = [(r, N')]@P$$

- Otherwise, Π is undefined.

Lemma 1 shows that the paths are preserved along reduction. It requires the potentials P and Q to be in the domain of the projection, this condition is the counterpart of the "long enough" condition on paths in GoI settings [5].

Lemma 1. *If $T, U \in Tra$, $P, Q \in Pot^N$, $\Pi(P) = P'$ and $\Pi(Q) = Q'$ then $(P, T) \mapsto^* (Q, U) \Rightarrow (P', T) \mapsto^* (Q', U)$*

If $T, U \in Tra$, $P', Q' \in Pot^{N'}$ and $(P', T) \mapsto^+ (Q', U)$ then there exists P, Q such that $\Pi(P) = P'$, $\Pi(Q) = Q'$ and $(P, T) \mapsto^+ (Q, U)$

In particular, the successive projections of free ports of a net will always be defined along a reduction sequence. So a path between two free ports of a net will always be stable along reduction, as stated by Corollary 1.

Corollary 1. *If $M \to^* N$, $p, q \in P_f^M$ and $T, U \in Tra$, then*

$$([(\overline{p}, M)], T) \mapsto^* ([(q, M)], U) \Leftrightarrow ([(\overline{p}, N)], T) \mapsto^* ([(q, N)], U)$$

Let Π_1, Π_2, Π_3 and Π_4 be the projections corresponding to the reduction steps of Figures 1 and 4. We have $\Pi_1([(e_0, N); (r_1, R_{\delta,\epsilon})]) = [(e_0, N_1); (r_1, R_{\delta,\epsilon})]$, then $\Pi_2([(e_0, N_1); (r_1, R_{\delta,\epsilon})]) = [(h_1, N_2)]$, next $\Pi_3([(h_1, N_2)]) = [(i_2, N_3)]$ and $\Pi_4([(i_2, N_3)])$ is not defined.

We can observe the reductions of the path of N $([(e_0, N); (r_1, R_{\delta,\epsilon})], []) \mapsto^{13} ([(e_0, N); (e_2, N)], [])$ which becomes $([(e_0, N_1); (r_1, R_{\delta,\epsilon})], []) \mapsto^2 ([(d_2, N_1)], [\overline{(\delta, 1)}]) \mapsto^4 ([(\overline{a_2'}, N_1)], [(\zeta, 2)]) \mapsto^5 ([(e_0, N); (e_2, N)], [])$ in N_1, then $([(h_1, N_2)], []) \mapsto^3 ([(i_2, N_2)], [])$ in N_2 and finally $([(i_2, N_3)], []) \mapsto^0 ([(i_2, N_3)], [])$ in N_3.

IV. Context semantics for complexity bounds

In this section, we define *canonical cells*, which are the potential ports which correspond to cells that will really appear during reduction. Then we use the canonical cells to define a weight $W_N \in \mathbb{N} \cup \{\infty\}$ for any net N such that, if $M \to N$, then $W_M \geq W_N + 1$. It follows that the length of any reduction

sequence from M is bounded by W_M. Notice that it is not true that $W_M > W_N$ because if $W_M = \infty$, then $W_N = \infty$.

The approach is inspired by Dal Lago's context semantics for linear logic [3]. First, Dal Lago's weight allowed to show that every proof-net of some linear logic subsystems verified complexity properties (e.g. every proof-net of LLL reduce in polynomial time w.r.t the size of the argument, whatever the reduction strategy). These bounds were previously known, but Dal Lago's proofs were much shorter. Then, his tool was used to prove strong bounds which were previously unknown [15], [2]. We hope that our tool will lead to similar results.

We want to capture the "cells which will appear during reductions beginning by N". Such a cell is either a cell of N, or appears during the reduction of two cells c_1 and c_2 such that: c_1 and c_2 both appear during reductions beginning by N, and $\{c_1, c_2\}$ will form an active pair. This is the intuition behind the following definition of canonical cells.

Definition 5. *We define the set Can^N of canonical cells of N by induction:*

- *For every cell c of N, $[(p_0(c), N)]$ is a canonical cell*
- *If $P_1.(p_0(c_1), N_1) \in Can^N$, $(P_1.(\overline{p_0(c_1)}, N_1), []) \mapsto (P_2.(p_0(c_2), N_2), [])$, $l^N(c_1) = s_1$, $l^N(c_2) = s_2$ and $L(s_1, s_2)$ is defined. Then for every cell c of N_{s_2,s_1}, $P_1.(p_0(c_1), N_1).(p_0(c), N_{s_2,s_1}) \in Can^N$.*

Lemma 2. *Let us suppose that $N \to_L N'$ by reducing the active pair $\{c_1, c_2\}$ and Π is the associated projection.*

If $P \in Can^N$, then either $\Pi(P)$ is defined and $\Pi(P) \in Can^{N'}$ or P corresponds to one of the ports of the active pair: $P \in \{[(p_0(c_1), N)], [(p_0(c_2), N)]\}$.

If $\Pi(P)$ exists and is in $Can^{N'}$, then $P \in Can^N$.

As an example, let us consider the net N of Figure 1. We can show that $C_1 = [(e_0, N); (e_1, R_{\delta,\epsilon})]$ is a canonical cell. Indeed, b_0 is a principal port of N so $[(b_0, N)]$ is a canonical cell. We know that $([(\overline{b_0}, N)], []) \mapsto^0 ([(a_0, N)], [])$ and $L(\zeta, \delta)$ is defined so $[(b_0, N); (d_2, N_{\delta,\zeta})]$ is a canonical cell. Finally, $([(b_0, N); (\overline{d_2}, N_{\delta,\zeta})], []) \mapsto^1 ([(e_0, N)], [])$ and $L(\delta, \epsilon)$ is defined so $[(b_0, N); (d_2, N_{\delta,\zeta}); (e_1, R_{\epsilon,\delta})]$ is canonical.

Similarly, $C_2 = [(a_0, N); (d_2, N_{\zeta,\delta}); (e_1, N_{\epsilon,\delta})]$ and $C_3 = [(e_0, N); (e_1, N_{\delta,\epsilon})]$ are canonical. Let Π_1, Π_2, be the projections corresponding to $N \to N_1$ and $N_1 \to N_2$ (Figures 1 and 4). We can observe that $\Pi_2 \circ \Pi_1(C_1) = \Pi_2 \circ \Pi_1(C_2) = \Pi_2 \circ \Pi_1(C_3)$ so, intuitively, there are three canonical cells corresponding to the same future cell.

The following theorem corresponds to the main result of [3]. The intuition behind it is that each reduction step erases two canonical potentials: the ones corresponding to the active pair.

Theorem 1. *For every interaction-net N, the length of any interaction sequence beginning by N is bounded by :*

$$T_N = \sum_{P \in C^N} \frac{1}{2^{|P|}}$$

V. Conclusion

We build, for interaction nets, a tool similar to the linear logic context semantics. Interaction nets being quite general, this tool could help to prove strong bounds for various systems.

In [15], we defined abstract properties based on Dal Lago's context semantics, and proved that if all the proof-nets of a linear logic subsystem verify the properties, the system is polynomially sound. Such criteria could be especially useful in interaction nets. For example, we may think that LLL (subsystem of linear logic), $DLAL$ (λ-calcul type system based on LLL) and LPL (type system for λ-calculus extended with pattern matching based on $DLAL$) satisfy a common property which we could express in terms of context semantics. Thus, we could prove the bounds for those three systems in a uniform way. This may ease the transformation of other linear logic subsystems (SLL, $QBAL$, L^4) into programming languages.

However it seems we need further tools (corresponding to the notion of copies, acyclicity of proof-nets and subtree properties in [3]) to ease the use of Theorem 1 to prove bounds for interaction nets system. This is left for future work.

We also used this CS to build a denotational semantics for interaction net systems.

References

[1] P. Baillot and M. Pedicini. Elementary complexity and geometry of interaction. *Fundamenta Informaticae*, 45(1-2):1–31, 2001.

[2] Patrick Baillot, Paolo Coppola, and Ugo Dal Lago. Light logics and optimal reduction: Completeness and complexity. *Information and Computation*, 209(2):118–142, 2011.

[3] U. Dal Lago. Context semantics, linear logic, and computational complexity. *ACM Trans. Comput. Log.*, 10(4), 2009.

[4] V. Danos and L. Regnier. Proof-nets and the Hilbert space. *London Mathematical Society Lecture Note Series*, pages 307–328, 1995.

[5] Marc De Falco. Géométrie de lâĂŹinteraction et réseaux différentiels. *These de doctorat, Université Aix-Marseille*, 2, 2009.

[6] Maribel Fernández, Ian Mackie, Shinya Sato, and Matthew Walker. Recursive functions with pattern matching in interaction nets. *Electronic Notes in Theoretical Computer Science*, 253(4):55–71, 2009.

[7] J.Y. Girard. Proof-nets: the parallel syntax for proof-theory. *Logic and Algebra*, 180:97–124, 1996.

[8] G. Gonthier, M. Abadi, and J.J. Lévy. The geometry of optimal lambda reduction. In *Proceedings of the 19th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pages 15–26. ACM, 1992.

[9] Georges Gonthier, Martín Abadi, and J-J Lévy. Linear logic without boxes. In *Logic in Computer Science, 1992. LICS'92., Proceedings of the Seventh Annual IEEE Symposium on*, pages 223–234. IEEE, 1992.

[10] Y Lafont. Interaction nets. In *Principles of programming languages, 17th ACM SIGPLAN-SIGACT symposium on*, pages 95–108. ACM, 1989.

[11] U Dal Lago. The geometry of linear higher-order recursion. In *Logic in Computer Science, 2005. LICS 2005. Proceedings. 20th Annual IEEE Symposium on*, pages 366–375. IEEE, 2005.

[12] Sylvain Lippi. Encoding left reduction in the lambda-calculus with interaction nets. *Mathematical Structures in Computer Science*, 12(6):797–822, 2002.

[13] Ian Mackie and Jorge Sousa Pinto. Encoding linear logic with interaction combinators. *Information and Computation*, 176(2):153–186, 2002.

[14] Damiano Mazza. Multiport interaction nets and concurrency. In *CONCUR 2005–Concurrency Theory*, pages 21–35. Springer, 2005.

[15] M. Perrinel. On paths-based criteria for polynomial time complexity in proof-nets (long version). http://arxiv.org/abs/1201.2956, 2013.